```c
//  1. Write a program to read and display the content of a text file.

#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>

#define BUF_SIZE 1000

int main(int argc,char *argv[]){
    int fd1 = open(argv[1],O_RDONLY);
    int size;
    char buf[BUF_SIZE];
    while((size=read(fd1,buf,BUF_SIZE)) >0 )
    {
        write(1,buf,size);// writing on to the terminal
    }

}

//  2. Write a program to copy the contents of one file to another.

#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>

#define BUF_SIZE 1000

int main(int argc,char *argv[]){
    int fd = open(argv[1],O_RDONLY);
    int fd2 = open(argv[2],O_WRONLY|O_CREAT,0644);
    int len;
    char buffer[BUF_SIZE];
    while((len=read(fd,buffer,BUF_SIZE)) >0)
    {
        write(fd2,buffer,len);
    }

}

//  3. Implement rm command using suitable file APIs


#include<unistd.h>
#include<stdio.h>
int main(int argc,char *argv[]){
    unlink(argv[1]); // same as rm ; to remove the file
}

//  4. Write a program to create a process and print its id and its parent id.

#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>

int main(){
    int i=fork(); // forking a process
    printf("Forking process ....\n");
    printf("Getting process id ...\n");
    printf("PID : %d PPID : %d \n",getpid(),getppid());
    return 0;
}


//  5. Write a program to implement the following command
//  cat file1 file2 >> dest_file

#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#define BUF_SIZE 100

int main(int argc,char *argv[])
{
    if(argc!=4)
```

```
    {
        printf("Invalid Number of arguements!");
        return 1;
    }
    char buf[BUF_SIZE];
    int size;
    int fd1=open(argv[3],O_WRONLY|O_CREAT|O_APPEND);
    int fd=open(argv[1],O_RDONLY);
    while((size=read(fd,buf,BUF_SIZE)))
    {
        write(fd1,buf,size);
    }
    close(fd);
    fd=open(argv[2],O_RDONLY);
    while((size=read(fd,buf,BUF_SIZE)))
    {
        write(fd1,buf,size);
    }
    return 0;
};



//  6. Write a program to read a string input from user. Append it to an existing file. If file is not
//  existing, then it need to be created. After the end of input display the file contents.

#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#define BUF_SIZE 1000

int main(int argc,char *argv[])
{
    if(argc!=2){
        printf("Invalid Number of arguements!");
        return 1;
    }
    printf("Press Enter to start typing the string (Ctrl+d to exit): ");
    getchar();
    int fd=open(argv[1],O_WRONLY|O_APPEND|O_CREAT,0644);
    char cont[BUF_SIZE];
    int size;

    while(size=read(0, cont,BUF_SIZE))
    {
        write(fd,cont,size);
    }
close(fd);
return 0;
}

//  7. Write a program to print the file user id and inode no. of a file. The name of the file needs to
//  be passed as command line argument. Show suitable message, if the file does not exist.

#include<stdio.h>
#include<sys/stat.h>
#include<fcntl.h>

int main(int argc,char *argv[]){
    if(argc!=2){
        printf("Invalid Number of arguements!");
        return 1;
    }
    struct stat st;
    int a=stat(argv[1],&st);
    if(a>=0)
        printf("\nInode number: %d\nUID: %d\n",st.st_ino,st.st_uid);
    else
        printf("Invalid file");
}

//  8. Write a C program to create a file with 16 bytes of arbitrary data from the beginning and
//  another 16 bytes of arbitrary data from an offset of 48. If the file with given name is exists,
then old
```

```c
//  contents need to be copied to another file before writing new data

#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#define BUF_SIZE 1000
int main(int argc,char *argv[])
{

    char file[50];
    int file_descriptor1=open(argv[1],O_CREAT|O_RDWR,0644);
    printf("%d",file_descriptor1);
    if(access(file_descriptor1,F_OK) >= -1)    {
        printf("\nInside IF\n");
      int fd_copy = open("task8_bak.txt",O_CREAT|O_WRONLY,0644);
      int size;
      while((size = read(file_descriptor1,file,50)) > 0){
        printf("%d",size);
        write(fd_copy,file,size);
      }
    }
    int file_descriptor2=open(argv[2],O_RDONLY);
    char buf[17]="1234567890123456";
    char buf2[17]="0123456789012345";
    write(file_descriptor1,buf,16);
    close(file_descriptor1);
    file_descriptor1=open(argv[1],O_CREAT|O_RDWR,0644);
    lseek(file_descriptor1,48,1);

    write(file_descriptor1,buf2,16);

    return 0;
}


//  9. What are exit handlers? Explain their working with a suitable C program example

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
void func1();
void func2();
int main(int argc,char *argv[])
{
    if(argc!=2)
    {
        printf("Invalid Number of arguements! Pass the parameter: \n");
        printf("1:exit\n");
        printf("2:_exit\n");
        return 1;
    }

    atexit(&func1);
    atexit(&func2);

    switch(atoi(argv[1]))
    {
    case 1:
        exit(0);
        break;
    case 2:
        _exit(0);
        break;
    default:
        printf("Invalid parameter\n");
        break;
    }
}
void func1(){
    printf("Func 1 exiting\n");
}

void func2(){
    printf("Func 2 exiting\n");
```

```
}
```